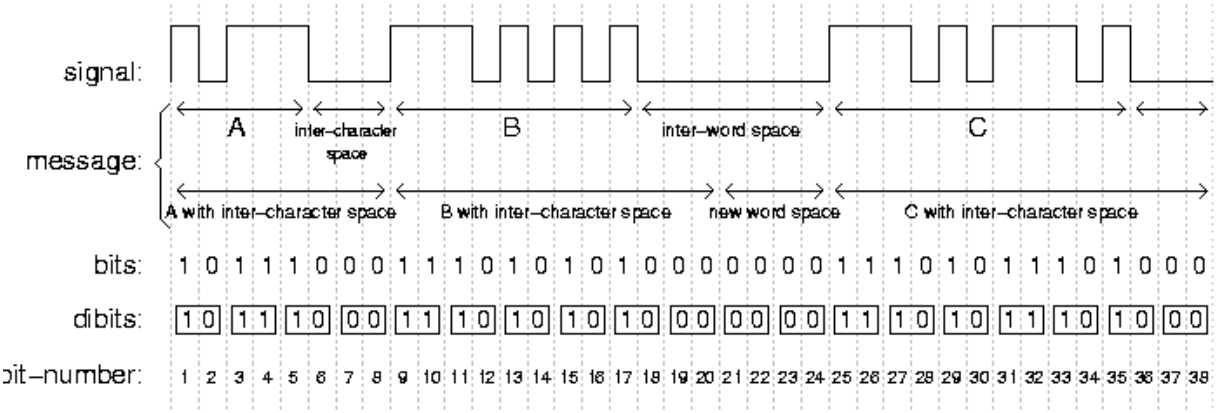
RSCW – Algoritam za prepoznavanje telegrafije

Iako se posljednje vrijeme sve češće susrećemo s automatskim dekodiranjem telegrafije i slanjem takvih podataka na Internet (poput CW skimmera), digitalna obrada signala u svrhu prepoznavanja telegrafije nije novost. Sigurno ste se zapitali kako to zapravo radi i na koji način računalo može prepoznati i dekodirati telegrafske signale. U nastavku ovog članka biti će opisan jedan od algoritama za prepoznavanje telegrafije i dani primjeri izvornog koda koji vam mogu biti dobar oslonac za razumijevanje ili pisanje vlastitog programa za dekodiranje Morseovog koda.



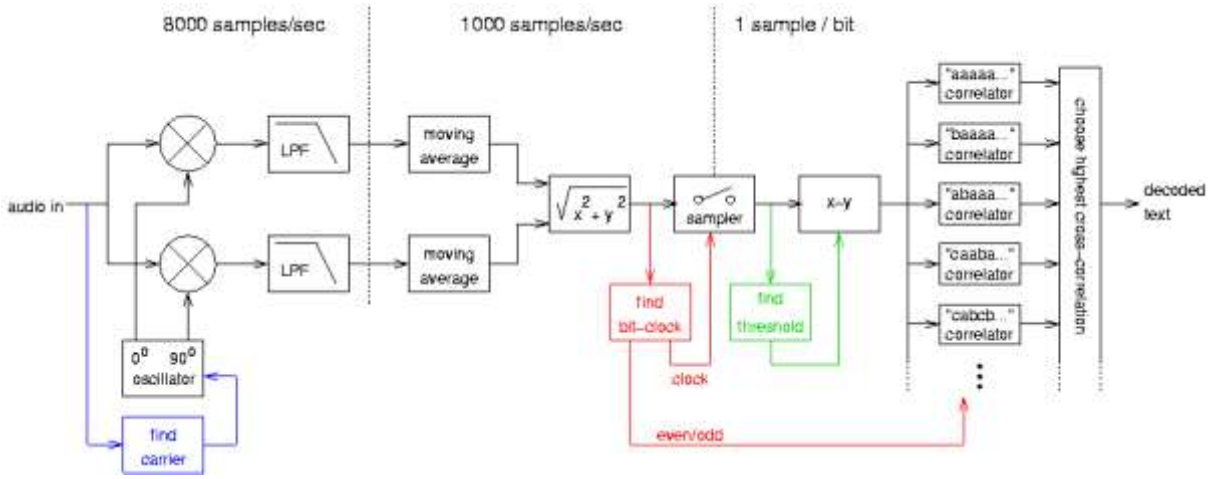
Algoritam

RSCW algoritam dobio je ime po engleskim riječima receive (primati) i soundcard (zvučna kartica). Njegov autor, Pieter (PA3FWM), razvio ga je kao pomoć za dekodiranje signala iz ruskog radioamaterskog satelita RS-12. Nije idealan: korisnik mora upisati približnu brzinu odašiljanih telegrafskih signala i dekodiranje ima neko vrijeme kašnjenja, ali s druge strane to mu daje mogućnost detekcije i vrlo slabih signala, a osim toga jednostavan je za razumijevanje i time odličan kao primjer.

Za početak, važno je razumijeti osnove telegrafije: crta traje tri puta duže od točke; razmak između crte i točke traje koliko i jedna točka; razmak između dva znaka traje tri dužine trajanja točke; razmak između riječi traje sedam dužina trajanja točke.

Prevodeći telegrafiju u računalni jezik, moramo sve svesti na 1 i 0 (binarni sustav, jedini kojega računalo razumije; 1 i 0 ćemo zvati bitovi). U slučaju ovog algoritma će 1 označavati vrijeme kada je odašiljač na predaji, a 0 vrijeme kada je odašiljač isključen. Tako bi, primjerice, slovo R za računalo bilo „1011101“ (točka-crta-točka, s uključenim već opisanim razmacima već između crte i točke). Na taj način možemo zaključiti sljedeće: točku predstavlja kombinacija „10“, crtu kombinacija „1110“, a na kraj svake od tih kombinacija dodajemo „00“ kako bi mogli jednoznačno razlikovati slova. Ovo znači da će svako slovo imati paran broj bitova.

Da bi bilo jednostavnije razumjeti algoritam, svaki par bitova predstavljen je slovima. „a“ je „00“, „b“ je „10“, „c“ je „11“, a „t“ je „00“.



Opis slike: crnim elementima prikazan je tok signala: signal ulazi s lijeve strane kao audio signal (uzorkovan na 8 bitova po uzorku, 8000 bitova u sekundi), a s desne strane algoritam daje dekodiranu poruku. Plavi, crveni i zeleni elementi odrađuju zadaće traženja frekvencije nosioca, pronalaženje vremenskog slijeda i pronalaženje nivoa signala.

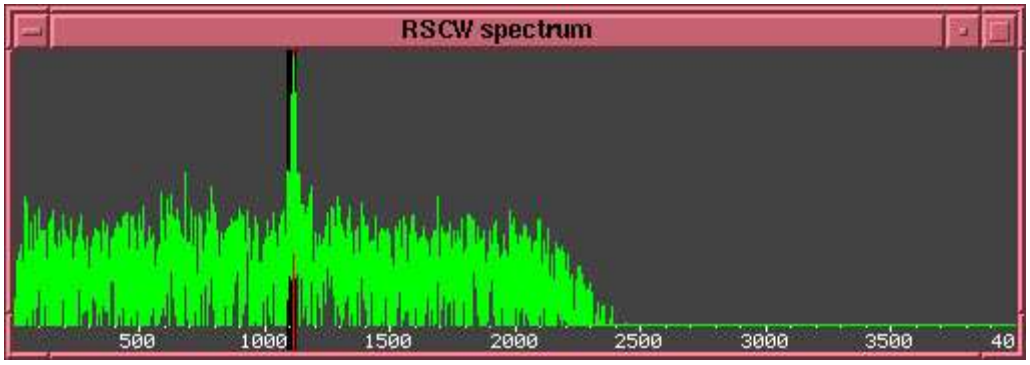
Dekodiranje signala

Dekodiranje se sastoji od sljedećih koraka:

1. Množenje ulaznog signala s lokalno generiranim signalom iste frekvencije kao što je ulazni signal. Ovim se kreiraju dva signala: jedan je ulazni signal koji se miješa sa samim sobom (ili približno istim signalom generiranim unutar dekodera) i čime se dobiva signal frekvencije blizu 0 Hz i drugi koji se miješa s generiranim signalom pomaknutim za 90°C. Nazvati ćemo te signale I (eng. In-phaze) i Q (eng. Quadrature)
2. Provlačenje oba signala kroz niskopropusni filter. Zbog miješanja signala iz prethodnog koraka, kroz ovaj će filter proći samo signali koji su izvorno bili vrlo blizu frekvencije nosioca – tj. ovaj se niskopropusni filter zapravo ponaša kao pojasnopropusni filter. Već smo spomenuli da dekoder radi s 8000 bita po sekundi. Nakon ovog filtera, imamo samo 1000 bita. Ovo je prihvatljivo, s obzirom da su uklonjene sve visoke frekvencije, a smanjuje se i opterećenje na ostali dio algoritma.
3. Računanje kretanja signala 1 bita, za I i Q signale. Signali koji dolaze u ovu fazu su neprekinuti niz bitova. Algoritam sada prepoznaje dali se radi o 0 ili 1 na određenom uzorku signala. Ovaj se korak radi zbog bolje detekcije slabih signala.
4. Kako je faza ulaznog signala još uvijek nepoznata (ne zna se što je 1, a što 0), računa se zbroj I i Q dijelova signala. Jednom kad se ova operacija napravi, faza ulaznog signala nije više važna. Rezultat ovoga je zelena crta prikazan na donjoj skici.
5. Za svaki bit uzima se uzorak iz gornje sume signala. Ovo se uzorkovanje mora uzeti u ispravnim trenucima – da budemo sigurni da se radi o periodu dok traju crta, točka ili pauza, a ne na njihovim prijelazima. Za uzimanje točnog trenutka brine se unutrašnji brojčanik.
6. Oduzimanje svakog uzorka od praga signala. Dobar prag signala negdje je između 0 (što označava isključen odašiljač) i 1 (što označava da je odašiljač uključen, ali signal sadržava i

šum). Rezultat ove operacije (idealan rezultata) je 1 ili 0. Međutim, kada imamo puno šuma, ovo ne mora biti točno. Ovaj je signal na grafu označen svjetlo plavom bojom.

7. Napokon, računamo odnos između uzoraka koje smo uzeli i slijedova nula i jedinica koji odgovaraju mogućim kombinacijama slova, brojki i znakova. Rezultat je najizglednija kombinacija detektiranih signala i mogućih kombinacija.



Traženje najizglednije poruke

Kad pričamo o „poruci“ najčešće mislimo o jednom slovu (iako to ne more biti slučaj), a izraz „poruka“ koristi se zbog općenitosti. Jedan od mogućih algoritama koristiti će parove bitova (jer smo već prije zaključili da sva slova, brojke i znakovi imaju paran broj bitova), a sastoji se od:

1. Započnemo s praznom porukom
2. Proširimo ju sa svim mogućim kombinacijama parova bitova (11, 10 i 00; dakle kreiramo tri poruke od početne jedne) i za svaku od njih radimo usporedbu s istim takvim parovima bitova primljenog signala i računamo faktor korelacije.
3. Tri dobivene poruke dodatno proširimo s mogućim kombinacijama parova bitova. Međutim, sada više nisu moguće sve kombinacije. Npr. nakon para bitova 11 ne može doći 00, nego samo 10. Koje su kombinacije moguće diktira sam morseov kod (npr. 6 uzastopnih crta nisu početak niti jednog znaka u morseovom kodu). I u ovom koraku ponovo računamo faktor korelacije.
4. Ponavljamo korak 3 do kraja emitiranja.

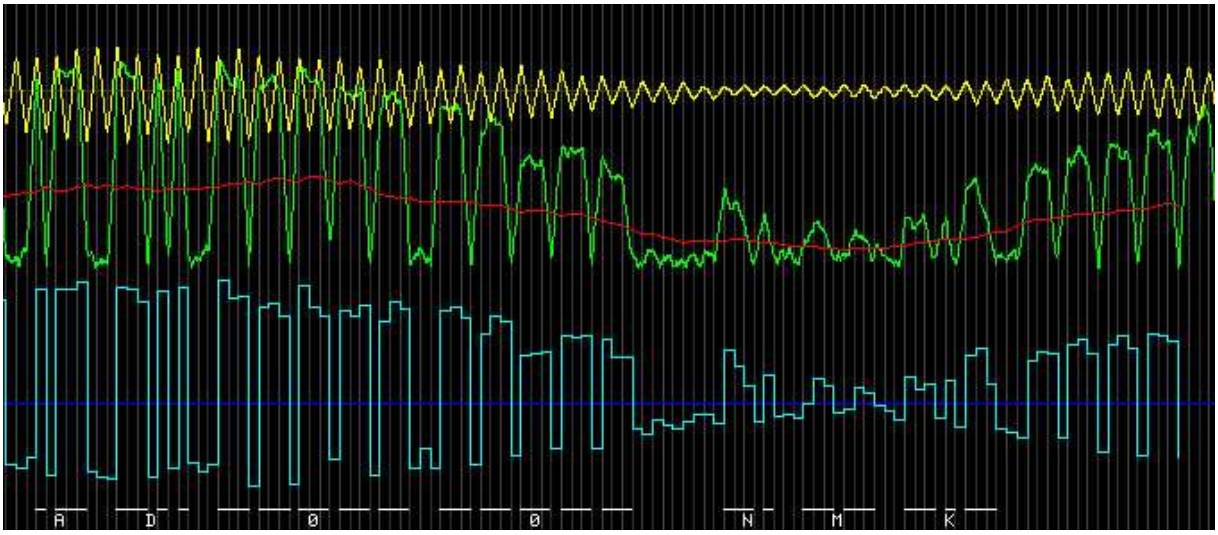
Očito je da se, nakon svakog ponavljanja koraka 3, broj poruka povećava s faktorom 1 i 3, ovisno o koliko je mogućih nastavaka. Ovo za posljedicu ima vrlo velik broj poruka što može dovesti do nepraktičnosti. Ovo je potrebno jer će se tek na kraju emitiranja detektirati najizglednija poruka, ali se ipak može malo pojednostaviti.

Pretpostavimo da nakon prijema 8 bita imamo sljedeće 3 poruke pohranjene (imamo ih više, ali ćemo uzeti samo tri za primjer):

10101000 – ovo je točka, točka, točka: „S“ s faktorom korelacije 3.4
11101000 – ovo je crta, točka: „N“ s faktorom korelacije 4.1
10111000 – ovo je točka, crta: „A“ s faktorom korelacije 2.3

Svaka od gornjih poruka predstavlja potpun morseov kod zajedno s dodanim „00“ na kraju, koje označavaju kraj slova. Svaka od ovih poruka može biti proširena s istim parom bitova. To znači da će

svaki od nastavaka koje sljedeće primimo biti mogući nastavak na svaku od ove tri poruke, a faktor korelacije će i nakon dodavanja novog para bitova biti isti (upravo zato što su sva tri nastavka moguća na zadnji par bitova „00“). Drugim riječima: prva i treća poruka niti nakon prijema sljedećeg para bitova neće imati veći korelacijski faktor od druge poruke, pa ih možemo zanemariti i ostaviti samo drugu. Zaključak je sljedeći: kada naiđemo na par bitova „00“, što nam označava kraj slova, zadržimo samo najizgledniju poruku, tj. poruku s najvećim korelacijskim faktorom. S ovime smo dobili i mogućnost da ne moramo čekati kraj odašiljanja da možemo ispisati primljenu poruku, već možemo ispisivati kako nailazimo na kraj slova.



Traženje frekvencije nosioca

RSCW algoritam koristi jednostavnu metodu za to: napravi se FFT (eng. Fast Fourier Transform) na segmentima ulaznog signala od 1 sekunde, te odabire frekvenciju na kojoj je signal najjači (raspon frekvencija zadaje korisnik). Ovo se radi dva puta u jednoj sekundi. Linearna interpolacija koristi se za pronalaženje frekvencije između dva uspješna rezultata FFT analize.

Algoritam radi dobro dokle god se radi o samo jednom signalu u zadanom frekvencijskom rasponu. U protivnom, mogao bi se lako „seliti“ iz jednog signala na drugi, što bi rezultiralo pogrešnim i neupotrebljivim dekodiranjem.

Određivanje praga signala

Na uzorku od 10 bitova (5 bitova prije i 5 nakon referentnog bita) odabire se takav prag signala čija je vrijednost između prosjeka najjačih primljenih signala i prosjeka najtiših primljenih signala. Ako bi ti prosjeci bili isti, uzela bi se srednja vrijednost. Ako se prijemni signal sastoji od puno točaka, tada je za očekivati da će algoritam za prag postaviti visoku vrijednost. Međutim, da bi se tome doskočilo, algoritam prati promjene i u takvim slučajevima uzima razliku najjačega i najtišega signala, a ne njihovih prosjeka. Signal koji se analizira gleda se na slijedeći način: ako je viši od praga prelazi u stanje 1, a ako je niži prelazi u stanje 0.

Program

Izvorni kod pisan je u programskom jeziku C i za obradu signala koristi standardne datoteke koje dolaze s razvojnim alatom, a i dio su većine linux distribucija. Zbog veličine koda, nije prikladan za objavljivanje.

Autor algoritma pripremio je i program u kojega je ugradio ovaj algoritam, zajedno s prikazom dekodiranih poruka. Program se može skinuti sa stranice:http://wwwhome.cs.utwente.nl/~ptdeboer/ham/rscw/

Paket se sastoji od nekoliko programa:

- rscw: glavni program za dekodiranje telegrafije. Program čita zvučni signal iz audio ulaza (stdin), s 8000 uzoraka u sekundi, što je ujedno i pretpodešena vrijednost za /dev/dsp uređaj na linuxu (napomena: program je pisan u C programskom jeziku i primjenjiv je na bilo kojoj platformi). Dekodirana poruka ispisuje se na ekran ili u datoteku.
- rscwx – pomoćni program koji pokreće rscw program u grafičkom sučelju, u kojemu crta grafove i analizira signale